Experiments with a Gaussian Merging-Splitting Algorithm for HMM Training for Speech Recognition*

Ananth Sankar

Speech Technology and Research Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

ABSTRACT

It is well known that the expectation-maximization (EM) algorithm, commonly used to estimate hidden Markov model (HMM) parameters for speech recognition, is sensitive to the initial model parameter values, making appropriate parameter initialization important. We investigate the use of iterative Gaussian splitting and EM training to initialize the desired number of Gaussians per HMM state (or state cluster). We then study merging of Gaussians which contain little training data as an approach to robust parameter estimation. Finally Gaussian merging and splitting is combined to form the Gaussian Merging-Splitting (GMS) algorithm. Detailed experimental studies show that Gaussian splitting gives similar performance to our previous training algorithm, even though the two algorithms give very different parameter values. The robust parameter estimation from Gaussian merging results in better performance than our old algorithm for speaker-independent models that have a large number of parameters relative to the amount of training data. In addition, in one experiment, speaker adaptation gave a 7% relative improvement in word error rate over the SI models when the SI models were trained with Gaussian splitting alone, as compared to a 15.5% improvement when the SI models were trained with both splitting and merging, even though the unadapted SI models gave similar performance. Finally, experiments with the GMS algorithm show that, for a given number of Gaussian parameters, better performance is achieved by reducing the number of HMM state clusters and increasing the number of Gaussians per state cluster.

1. Introduction

Most conventional automatic speech recognition (ASR) systems are based on context-dependent (CD) phone-based hidden Markov models (HMMs) that use Gaussian mixture models (GMMs) for the state-conditioned observation densities. A commonly used CD unit is the triphone, which is a model of a phone in the context of left and right phones. The number of observed triphones in the training data is usually very large, with many triphones having very little training data, resulting in poor estimates of the model parameters. One solution to this problem is to use HMM state clustering where the states in a cluster share a set of parameters, such as a set of Gaussians [1].

The HMM parameters are usually computed by maximum-likelihood (ML) estimation using the expectation-maximization (EM) algorithm [2]. The EM algorithm is an iterative procedure that recomputes the model parameters given their current estimates so as to increase the likelihood of the training data at each iteration. This algorithm is sensitive to the values of the initial parameters, and guarantees only a locally optimal solution. Different systems use different approaches to initialize and train the model parameters. However, to the author's knowledge, there has been no clear comparative description in the literature of the effect of different initialization and training procedures on speech recognition performance. Such a description is of value for anyone training an HMM based speech recognition system. We believe the studies reported in this paper provide useful insight into the issue of HMM parameter initialization and training.

In this paper, we present the Gaussian Merging-Splitting (GMS) algorithm for HMM training. In this method, iterative Gaussian splitting and EM training is used to initialize the required number of Gaussians in each HMM state cluster. Starting with a single Gaussian, Gaussian splitting is used to increase the number of Gaussians at each stage of training until the required number of Gaussians is reached. In addition, at each stage, Gaussians are iteratively merged until each Gaussian has a minimum amount of training data. The GMS algorithm results in a variable number of Gaussians in each HMM state cluster. The number of Gaussians is automatically decided subject to a constraint on the maximum number in each state cluster. Detailed experimental results show the effect of Gaussian splitting and merging, and compare the performance with our previous training algorithm.

2. SRI's Previous Training Approach

SRI's DECIPHER^{TM} speech recognition system is based on HMM state clustering where the states in each cluster share the same set of Gaussians or Genone [1]. Each state in a cluster has a different mixture weight distribution to these shared Gaussians. The HMM states are clustered separately for each phone. This is done by first training a phonetically tied mixture (PTM) system, where all states in a phone share the same set of 100 Gaussians. The states in this phone are then clustered using bottom-up agglomerative clustering. For clustering, the distance between two states is given by the

*This work was sponsored by DARPA through the Naval Command and Control Ocean Surveillance Center under contract N66001-94-C-6048.

weighted-by-counts increase in entropy of the mixture weight distribution (to the shared 100 Gaussians) due to merging the two states [1]. The mixture weight distribution of the merged state is easily computed as the weighted-by-counts average of the individual distributions.

The Gaussians in each state cluster are initialized using the corresponding 100 PTM Gaussians. The 100 Gaussians in each phone are used to initialize the required number for each state cluster through a series of steps involving the selection of the most likely Gaussians for each state cluster, and also Gaussian merging. Details of the algorithm can be found in [1].

This approach poses a potential problem for the initial values of the Gaussians in the state clusters and hence the final models. The 100 PTM Gaussians cover the entire acoustic space for a particular phone. Each state cluster for this phone covers only a small part of this large acoustic space. Thus, the PTM Gaussians may not be appropriate for initializing the Gaussians in the individual state clusters, and may result in inefficient use of the parameters. To address this issue, we studied an initialization algorithm based on Gaussian splitting.

3. Gaussian Splitting

We implemented a new initialization scheme based on the splitting strategy commonly used in vector quantization [3]. In this approach, we first estimate a single Gaussian model for each Genone. Given the segmentation of data into HMM states, the ML estimate of these (single) Gaussians is globally optimal. We then split the Gaussian for each Genone into two by slightly perturbing the mean of the Gaussian along the direction of the standard deviation vector, and reestimate the model by further EM training. This process of splitting and retraining is repeated until the required number of Gaussians is achieved. At each stage, we can choose how many Gaussians to split. Thus, if there are currently n Gaussians that we want to increase to m Gaussians, then we split the $m-n$ Gaussians having the largest average variance. This average, computed by using the geometric mean, is a measure of the likelihood of the training data modeled by that single Gaussian. The Gaussian with the largest variance is the one for which the training data likelihood is minimum. Since our goal is to maximize the training data likelihood, splitting this Gaussian is intuitively appealing. Gaussian splitting is also used in the Cambridge University HTK system [4]. However, in that system, the Gaussian with the largest amount of data is split as opposed to the one with the largest variance. We do not believe that this difference will significantly affect the final recognition results.

Let the mean and variance of Gaussian i in a Genone be denoted as $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$, respectively. Let the parameters for the Gaussians resulting from splitting Gaussian i be denoted as $\boldsymbol{\mu}_{i,1}, \boldsymbol{\sigma}_{i,1}^2, \boldsymbol{\mu}_{i,2}$, and $\boldsymbol{\sigma}_{i,2}^2$. Then

$$\boldsymbol{\mu}_{i,1} = \boldsymbol{\mu}_i + \epsilon * \boldsymbol{s} \quad (1)$$

$$\boldsymbol{\mu}_{i,2} = \boldsymbol{\mu}_i - \epsilon * \boldsymbol{s} \quad (2)$$

$$\boldsymbol{\sigma}_{i,1}^2 = \boldsymbol{\sigma}_{i,2}^2 = \boldsymbol{\sigma}_i^2 \quad (3)$$

where $\boldsymbol{s}$ is a vector whose kth component is the standard deviation of the kth dimension of the original Gaussian, i, and ϵ is a small positive number. We set ϵ to 0.001 in our experiments.

Each state q in a state cluster has a separate mixture weight distribution to the Gaussians in that Genone, resulting in a Gaussian mixture model for the state given by $\sum w_i^q N(.; \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$. When we split Gaussian i, the mixture weight w_i^q of state q for Gaussian i is divided equally into the mixture weights for the resulting Gaussians.

The Gaussian splitting initialization approach can be configured in a variety of ways. For example, we may split all Gaussians at each stage, or may split only the single largest variance Gaussian, or may do something in between these extremes.

4. Gaussian Merging

If there is too little training data segmented into an HMM state cluster, then the Gaussians in the corresponding Genone will not be well estimated. To ensure robust Gaussian estimation, we used a Gaussian merging algorithm. In this method, the Gaussians in a Genone are iteratively merged using bottom-up agglomerative clustering until all Gaussians have at least a threshold amount of data. The optimum value of this threshold is experimentally determined. Gaussian merging results in an automatically selected variable number of Gaussians per Genone. The clustering distance we used between two Gaussians, N_1 and N_2, is given by the weighted-by-counts increase in entropy due to merging the Gaussians:

$$D(1,2) = \frac{(T_1+T_2)}{2}\log|C_{1,2}| - \frac{T_1}{2}\log|C_1| - \frac{T_2}{2}\log|C_2|, \quad (4)$$

where T_1 and T_2 are speech data counts, and $|C_1|$, $|C_2|$ and $|C_{1,2}|$ are the determinants of the covariance matrices for the individual Gaussians, and the merged Gaussian, respectively. $C_{1,2}$ can easily be computed from the sufficient statistics of the individual Gaussians.

5. The GMS Algorithm

In the GMS algorithm, Gaussian merging is done before each Gaussian splitting operation. This guarantees that at all stages of the algorithm, the Gaussians are robustly estimated for all Genones. In the GMS algorithm, the user must specify the number of Genones and the maximum number of Gaussians per Genone. The GMS algorithm iteratively increases the number of Gaussians using merging and splitting until the

maximum number of Gaussians is reached. Since the amount of training data per Genone varies, and Gaussians are merged until all Gaussians have a threshold of data, the number of Gaussians per Genone usually varies with the Genones after a certain number of merging and splitting operations. Genones with lesser training data have fewer Gaussians and vice versa. The number of Gaussians that are split thus varies with the Genones as the algorithm progresses.

6. Experimental Results

6.1. Data Description

We conducted detailed experiments using the Wall Street Journal (WSJ) corpus to study the effect of Gaussian splitting and merging, comparing performance with our previous training algorithm. For training, we used the WSJ SI-284 corpus, which consists of 142 male and 142 female speakers, with about 18,000 utterances per gender. For our experiments, we used half of the male speakers and 49 utterances per speaker for a training set of 3479 utterances. Since we ran numerous training experiments in this study, we used this smaller training set to reduce the time for each experiment. However, we believe the results will extend to larger training sets.

For recognition experiments, we used three test sets. The first consists of 10 male speakers taken from the 1993 WSJ development and evaluation test sets, each uttering about 23 sentences for a total of 230 sentences and 3645 words. This set is referred to subsequently as WSJ93. A 20,000-word bigram language model was used for this set. The second test set was the 10-speaker male subset of the 1994 WSJ S0 development test set and used a 5000-word bigram language model. This set consisted of 209 sentences with 3330 words, and is referred to as WSJ94S0. The third test set was the male subset of the 1995 North American Business News H3-C0 (Sennheiser microphone) development set and used a 60,000-word bigram language model. This set had 152 sentences with 3904 words, and is referred to as NABN95. All recognition experiments were run using word lattices [5] that we had previously generated for these test sets.

The speech features we used were 12 mel-frequency cepstral coefficients (MFCCs) together with their first- and second-order time derivatives, and the normalized energy along with its first- and second-order derivatives to give a 39-dimensional speech feature vector.

6.2. Effect of Gaussian Splitting

To estimate how well the Gaussians in a Genone model the training data, we can measure the number of feature vectors in each Gaussian. If some Gaussians model most of the data, and the others only a few data points, it might be an indication of inefficient parameter usage caused by poor initial conditions. Let n_i denote the number of feature vectors in the ith Gaussian of some Genone, and $N = \sum n_i$ be the total number of feature vectors for that Genone. The entropy of the distribution $p_i = n_i/N$ gives an estimate of how the data is distributed into the Gaussians. The entropy is computed as

$$H = -\sum_{i=1}^{M} p_i \log p_i, \tag{5}$$

where M is the number of Gaussians in the Genone. The maximum value of this entropy is $\log M$, and it is achieved when the data is equally distributed into the Gaussians.

We trained a Genone-based HMM using the old initialization approach (Section 2) and the Gaussian splitting approach (Section 3). These models had 991 HMM state clusters, with each cluster sharing a Genone with 32 Gaussians. Thus, the maximum entropy for each Genone is 5 ($\log_2 32$). As explained previously, the Gaussian splitting approach can be configured in a variety of ways. For example, we may split all Gaussians at each stage, or may split only the single largest variance Gaussian, or may do something in between these extremes. We experimented with many of these approaches. There was not a very significant difference in performance between these methods, and so we decided on a simple strategy that splits all Gaussians at each stage until we have the desired number of Gaussians per Genone.

Figure 1 plots the entropy for each Genone in the HMM trained with our old initialization algorithm (see Section 2), and the solid line in Figure 2 plots the entropies for the Gaussian splitting approach. The x-axis is the Genone index, and the Genones are ordered from left to right on the axis according to an increasing amount of data per Genone. As we can see from the figures, the entropy for many Genones is much less than the maximum of 5 in the old approach, showing that for these Genones, the Gaussians are underutilized. However, the entropy for all Genones trained using the splitting approach is very close to 5. Clearly, the two initialization approaches have dramatically different influences on the model parameters.

From the figures, it appears that the Gaussian splitting approach does a better job of utilizing the model parameters. We ran recognition experiments using the old model and the one trained with Gaussian splitting. Table 1 shows the word error rates on the three test sets. As can be seen, Gaussian splitting gave essentially the same performance as the old approach. One possible problem with Gaussian splitting is that if there are not enough data points in a Genone, then each Gaussian may wind up getting only a small amount of data, since the data is equally distributed into the Gaussians. The Gaussian merging algorithm described in Section 4 should take care of this by merging Gaussians with too little data.

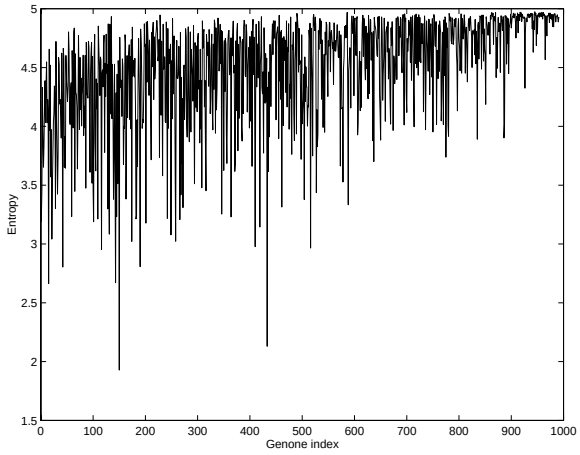

Figure 1: Entropy for Genones with the old training algorithm

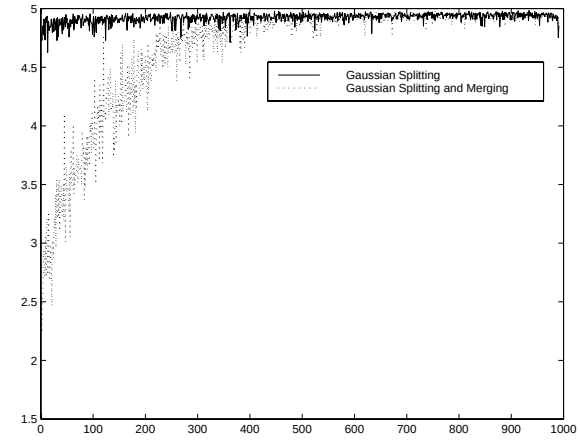

Figure 2: Entropy for Genones with the Gaussian splitting algorithm

Database	Word Error Rate (%)	
	Old Algorithm	Gaussian Splitting
WSJ93	23.7	24.0
WSJ94S0	13.7	13.9
NABN95	24.3	23.5

Table 1: Word error rates for different initialization algorithms

Database	Word Error Rate (%)				
	Old Algorithm	Gaussian splitting followed by merging Merging Threshold Values			
		0	25	50	100
WSJ93	23.7	24.0	23.9	23.5	23.9
WSJ94S0	13.7	13.9	13.5	13.5	13.8
NABN95	24.3	23.5	23.9	23.9	23.9

Table 2: Comparison of word error rates with different merging thresholds

6.3. Effect of Gaussian Merging

We merged the 32 Gaussians in each of the 991 Genones trained using the Gaussian splitting algorithm. This merging was followed by one iteration of EM training to reestimate the models, starting with the merged Gaussians. The dotted line in Figure 2 plots the entropy for the resulting Genones, showing that Genones with less data use fewer Gaussians while Genones with enough data use close to the maximum of 32 Gaussians. Table 2 shows the word error rates of the new training approach with different merging thresholds. It also replicates the word error rates obtained by using the old training approach.

From these results, we see that the merging algorithm resulted in a small improvement for WSJ93 and WSJ94S0. At a merging threshold of 50, the performance of the new algorithm is essentially the same as that of the old algorithm. Again, we see that no significant improvement in performance was achieved.

We investigated how speaker-independent (SI) models trained using the splitting and merging algorithms performed when they were adapted to the test speakers using maximum-likelihood transformation based adaptation where we used block-diagonal matrix affine transformations of the HMM mean vectors [6]. These experiments were carried out only on the WSJ93 set. Adaptation was done in supervised mode with the 40 common sentences provided by each speaker. Table 3 gives the word error rates for both the unadapted and adapted models. We can see clearly that Gaussian merging is critical to good adaptation performance. It is interesting that while the difference in the performance of the unadapted models is small, training the SI models by Gaussian splitting and merging resulted in a 15% adaptation improvement over the SI models as compared to only 7% when Gaussian splitting alone was used to train the SI models. This result can be explained by the greater robustness of the SI model parameter estimates from Gaussian merging as compared to splitting alone. Since the SI models are used to estimate the transformations used in adaptation, poor SI model estimates cause incorrect adaptation transformation estimates, which in turn result in poorly

Models	Training method for SI models Number of Genones = 991		
	Old	Gaussian Splitting	Splitting and Merging
Speaker-independent	23.7	24.0	23.5
Adapted	20.5	22.3	19.9

Table 3: Comparison of word error rates for WSJ93 before and after adaptation using different approaches to train the SI models

Models	Training method for SI models Number of Genones = 2027	
	Old	Splitting and Merging
Speaker-independent	25.3	23.9
Adapted	21.1	20.3

Table 5: Comparison of word error rates for WSJ93 before and after adaptation using different approaches to train the SI models

estimated adapted models. However, while Gaussian merging gave better performance than splitting alone, it is only slightly better than the old algorithm.

Table 3 clearly shows the effect of greater robustness due to Gaussian merging. We further investigated this by comparing the Gaussian splitting and merging algorithm to the old approach described in Section 2. Since the greater robustness of Gaussian merging would be more evident in much larger systems, we trained an HMM system with 2027 Genones and 32 Gaussians per Genone, using both the old training approach and the new approach (Gaussian splitting until each Genone has 32 Gaussians, followed by iterative Gaussian merging with a merging threshold of 50 frames). Table 4 shows the word error rates for this system, along with the corresponding word error rates for the smaller, 991-Genone system.

From this table, we see that for both the Gaussian splitting and merging and old training algorithms, the word error rate was higher with the larger system (2027 Genones). However, the system trained with the splitting and merging algorithm degraded more gracefully than the one trained with the old algorithm. The relative increase in word error rate for the old system on WSJ93, WSJ94S0, and NABN95 was 6.8%, 13.1%, and 7.0%, respectively, as compared to 1.7%, 4.3%, and 5%, respectively, for the new algorithm. Thus, the Gaussian splitting and merging training algorithm gives more robust estimates of the model parameters than the old algorithm. This greater robustness is reflected in the fact that the word error rates are not as sensitive to the number of model parameters, creating a wider range of the number of model parameters that give similar error rates. This makes it easier to experimentally search the space of models to find the optimal one.

Database	Word Error Rate (%)			
	Old algorithm		Gaussian Splitting and Merging Algorithm	
	991 Genones	2027 Genones	991 Genones	2027 Genones
WSJ93	23.7	25.3	23.5	23.9
WSJ94S0	13.7	15.5	13.5	14.1
NABN95	24.3	26.0	23.9	25.1

Table 4: Comparison of word error rates for systems with different numbers of parameters

Finally, in Table 5, we present adaptation results on WSJ93 for the 2027 Genone system when the SI models were trained with the old algorithm and the splitting and merging algorithm. The results show that the performance both before and after adaptation was superior with the splitting and merging algorithm.

7. Training with the GMS Algorithm

After studying the individual effects of Gaussian splitting and merging, we used the GMS algorithm to train Genonic HMMs by using the WSJ SI-284 subset described above. We experimented with different numbers of Genones and Gaussians per Genone. Table 6 gives the resulting recognition word error rates. Since the GMS algorithm guarantees robust parameter estimation, we experimented with large numbers of Gaussians per Genone. In previous development work, we had typically used 32 Gaussians per Genone. However, in using the GMS algorithm we decided to increase this upto 512 Gaussians per Genone. (To allow comparison across different settings with the same total number of Gaussian parameters, the total number of Gaussians is given in the table in parentheses.)

One interesting observation we can make from Table 6 is that the optimal system has 126 Genones and 512 Gaussians per Genone. This is significantly different from our previous systems, which typically used 1000 or more Genones and 32 Gaussians per Genone. We note, however, that we used only a fraction of the WSJ SI-284 data to train the systems listed in Table 6, and so the optimal number of parameters may be different from that of our previous systems. However, the table still shows a preference for systems with fewer Genones and more Gaussians per Genone. For example, comparing the 991-Genone/32-Gaussian system with the 126-Genone/256-Gaussian system, we see that the latter has a lower word error rate even though the number of parameters is compara-

Number of Genones / Test Set	Number of Gaussians per Genone 32	128	256	512
126 /		(16128)	(32256)	(64512)
WSJ93	-	23.7	23.0	23.0
WSJ94S0	-	12.9	12.3	11.7
NABN95	-	22.5	21.8	22.0
252 /		(32256)	(64512)	
WSJ93	-	22.8	22.5	-
WSJ94S0	-	13.2	12.5	-
NABN95	-	22.6	22.7	-
510 /	(16320)			
WSJ93	23.4	-	-	-
WSJ94S0	13.3	-	-	-
NABN95	23.8	-	-	-
991 /	(31712)			
WSJ93	23.8	-	-	-
WSJ94S0	13.2	-	-	-
NABN95	23.8	-	-	-

Table 6: Word error rates (%) with different number of Genones and Gaussians per Genone. The number of Gaussians is in parenthesis.

ble. In fact, the 126-Genone/128-Gaussian system also has a lower error rate than the 991-Genone/32-Gaussian system even though it has fewer parameters. One explanation for this might be that when two HMM state clusters overlap in acoustic space, modeling them as a single cluster with more Gaussians gives better resolution than modeling them as two clusters with half the number of Gaussians in each cluster.

Since the optimal models trained with the GMS algorithm have fewer Genones than our previous systems, there is the potential of a speedup for speech recognition in addition to the improved word error rate. At any frame during the Viterbi search, multiple hypotheses must be evaluated. For each hypothesis, the frame likelihoods of the Gaussians in the corresponding Genone are computed and cached. When the same Genone is used by another active hypothesis at the same frame, the cache is used to provide the likelihoods. For systems with fewer Genones, the same Genone is used for more of the active hypotheses at any frame, thus reducing the number of Genone likelihood computations. However, since the number of Gaussians per Genone is larger, each Genone likelihood computation is more expensive. We can reduce the cost of Genone likelihood computations by using Gaussian shortlists [1]. In this method, vector quantization is used to divide the acoustic space into Voronoi regions. A shortlist of Gaussians is maintained for each Genone in each region. During recognition, the Voronoi region corresponding to the test frame is used to determine the shortlist of Gaussians to be computed. By effectively using shortlists, we believe that a significant speedup can be achieved using the new models trained by the GMS algorithm. We are currently studying this issue.

8. Summary and Conclusion

We have introduced the GMS algorithm for HMM training and presented a detailed comparative study of the algorithm. Gaussian splitting has the property of uniformly distributing the data into the available Gaussians, while Gaussian merging results in robust parameter estimation. This robustness was shown by graceful degradation in performance when the number of parameters is larger than can be properly estimated by the available training data, and also by significantly improved performance of speaker adaptation algorithms. The GMS algorithm allows us to efficiently train HMMs by automatically determining the number of Gaussians for a given number of Genones. Since robust estimation is guaranteed, we can train systems with a very large number of Gaussians per Genone. Some interesting observations were made about the effect of varying the number of state clusters or Genones and the number of Gaussians per Genone. As a result, we found that the optimum HMM structure was very different from that used by our previous systems; the number of state clusters or Genones was significantly decreased while the number of Gaussians per Genone was significantly increased. The reduced number of Genones has the potential of giving a significant speedup during recognition.

References

1. V. Digalakis, P. Monaco, and H. Murveit, "Genones: Generalized Mixture Tying in Continuous Hidden Markov Model-Based Speech Recognizers," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 4, pp. 281–289, 1996.
2. A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.
3. Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, January 1980.
4. S. Young and P. Woodland, "The Use of State Tying in Continuous Speech Recognition," in *Proceedings of EUROSPEECH*, pp. 2203–2206, 1993.
5. H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub, "Large-Vocabulary Dictation Using SRI's DECIPHER(TM) Speech Recognition System: Progressive-Search Techniques," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. II 319–322, 1993.
6. L. Neumeyer, A. Sankar, and V. Digalakis, "A Comparative Study of Speaker Adaptation Techniques," in *Proceedings of EUROSPEECH*, pp. 1127–1130, 1995.